ICS 111
Final Review 2/2

- Classes
- Interfaces
- Objects
- References
- Methods and Parameters
- Inheritance
- Exceptions

ICS 111 review: Java Class types

- a class is the fundamental way of defining new types in Java
- a class typically includes instance variables and instance methods
  - an instance method is called from the object:

```
String s = new String("hello world");
s.substring(0, 5);   // returns "hello"
```

- classes often also provide static methods
  - a static method is called from the class:

```
Math.log(1.0);
```

- classes occasionally have static variables
- abstract classes can have abstract methods that have no implementation
- `instanceof` tells us whether it is safe to cast a class to one of its sub(sub-)classes

```
Object o = new String("hello world");
if (o instanceof String) s = ((String)o).substring(6);
```

ICS 111 review: Java Interfaces

- an interface lists a collection of public method headers
- each method header includes:
  - name
  - return type
  - parameter types
- all the methods in an interface are public
- a class may implement one or more interfaces
  - the compiler checks that all the methods listed in the interface are implemented in the class
- an interface can be used as the type of an object reference
  - in which case the object reference only provides the methods of the interface

ICS 111 review: Java Objects

- Object is the superclass of all other objects
  - meaning it is the root of the inheritance tree
  - and every Java object is an Object
- we have looked at instance methods toString, equals, and getClass
  - in each case, the method used is the one defined by the actual object, rather than by the class of the reference:

```
String s = new String("hello world");
Object o = s;
if (o.equals(s)) { // runs String.equals, not Object.equals
```

- other methods include:
  - hashCode: compute a (preferably unique) integer for this object
    - if `o1.equals(o2)`, then `o1.hashCode() == o2.hashCode()` should also be true
  - clone: make a copy of this object (protected method)
  - wait, notify, notifyAll: used for thread synchronization
-
-

ICS 111 review^2: Java References

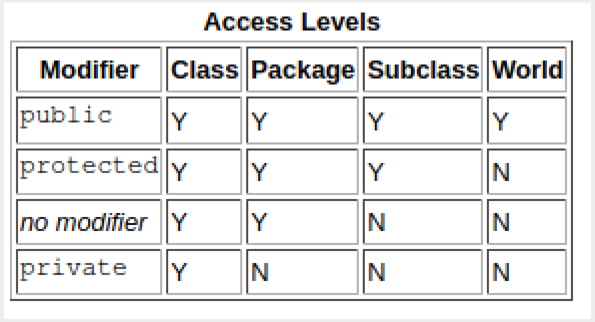
- every Object variable or Object-valued expression is a reference to the underlying object
- multiple references to the same underlying object are **aliases** for that one object:

```
int x[] = new int[10];
int[] y = x;
x[3] = 55;
if (y[3] == 55) { …  // true
```

- ultimately, a reference is a memory address
- special references: `null`, `this`, and `super`
-

ICS 111 review: Java Methods and Parameters

- a method header always has a return type, a method name, and a parameter list
- when implementing a method, the header is followed by the method body
  - in the body, the method parameters are local variables
- a method declaration can also be modified:
  - static (if not static, it is an instance method)
  - public, protected, private, (or default – package private)
- when calling (invoking) a method, must provide a value for each of the parameters
- for overloaded methods, Java selects the one that matches the parameters



Access Levels

Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

ICS 111 review: Inheritance

`class A extends B { ...`

means that `A` inherits all the (non-private) methods of `B`

- `A` can override any method inherited from `B`
  - name, parameters and return type must match exactly
- Java has **single inheritance**: a class can only extend one other class
  - but a class can implement any number of interfaces
- with single inheritance, all the classes can be arranged in a tree rooted at Object
- creating a superclass and several subclasses can be a way of having objects that are different but similar
- An object of a subclass type can have a reference of a superclass type:

```
Object o = new String("hello world");
```

- converting the other way requires a cast, and may throw a ClassCastException

```
String s = (String)new Object();
```

```
Exception in thread "main" java.lang.ClassCastException: class
java.lang.Object cannot be cast to class java.lang.String
```

- interfaces have their own hierarchy

ICS 111 review: Exceptions

- superclass Exception has many subclasses, including RuntimeException
- `throw new Exception("something wrong");`
- methods that may throw exceptions must declare that with `throws` (unless they only throw RuntimeException)
- code that throws an exception may be put between try and catch:

```
try {
  throw new StackOverflowError();
} catch (RuntimeError e) {
  System.out.println("caught exception " + e);
}
```

- can have one or more catch statements
  - the first one to match the exception is the one executed
- may end with a `finally` block

ICS 111 review: The big view

- ultimately we are trying to use computers to solve problems
  - calculate pi
  - write an exciting new game
  - keep track of addresses in a phone
- simple problems can be solved by simple programs
- complex problems require more complex programs
  - to have a chance at being correct, such programs have to be well structured
  - sometimes the focus is on the data (e.g. objects, classes)
  - sometimes the focus is on the program (e.g. methods, libraries)
  - most programs need attention to both
- methods, objects, class hierarchies, packages, and standard libraries are all ways to organize code and make programs more understandable
- complexity is also lessened when scope is limited (e.g. local variables) and access is limited (private/protected methods)
- programs are more understandable when we follow conventions